R E P O R T R E S U M E S

ED 020 128 SE 004 120

COMPUTER SCIENCES IN ELECTRICAL ENGINEERING.
COMMISSION ON ENGINEERING EDUC., WASHINGTON, D.C.

PUB DATE SEP 67

EDRS PRICE MF-$0.25 HC-$1.56 37P.

DESCRIPTORS- *CURRICULUM DEVELOPMENT, *COLLEGE SCIENCE, *ENGINEERING EDUCATION, *PHYSICAL SCIENCES, BIBLIOGRAPHIES, COMPUTER ORIENTED PROGRAMS, ELECTRONICS, INFORMATION SCIENCE, PHYSICS, UNDERGRADUATE STUDY, ASSOCIATION FOR COMPUTING MACHINERY, MATHEMATICAL ASSOCIATION OF AMERICA,

THE COMMITTEE ON COMPUTER SCIENCES IN ELECTRICAL ENGINEERING (COSINE COMMITTEE) OF THE COMMISSION ON ENGINEERING REPORTS ITS EXPLORATION OF THE ROLE OF ELECTRICAL ENGINEERING IN COMPUTER SCIENCES. GREATER FLEXIBILITY IN ENGINEERING CURRICULA IS FELT ESSENTIAL TO MEET THE EDUCATIONAL NEEDS IN SUCH A RAPIDLY CHANGING AND DIVERSE FIELD. THE MAJOR RECOMMENDATIONS INVOLVE THREE AREAS--(1) EDUCATION IN THE PRINCIPLES AND PRACTICES OF THE DESIGN OF GENERAL PURPOSE DIGITAL INFORMATION PROCESSING SYSTEMS, (2) SHIFT OF INFORMATION PROCESSING TECHNOLOGY FROM THE CONTINUOUS TO THE DISCRETE, AND (3) INTEGRATION OF DIGITAL COMPUTATION INTO MANY OF THE TRADITIONAL COURSES IN ELECTRICAL ENGINEERING CURRICULUM. APPENDIXES INCLUDE (1) SELECTED REFERENCES FOR BASIC SUBJECT AREAS, AND (2) EXISTING ELECTRICAL ENGINEERING CURRICULA OF SEVERAL SCHOOLS WITH A CONCENTRATION IN COMPUTER SCIENCE. (DH)

ED020128

Computer Sciences In Electrical Engineering

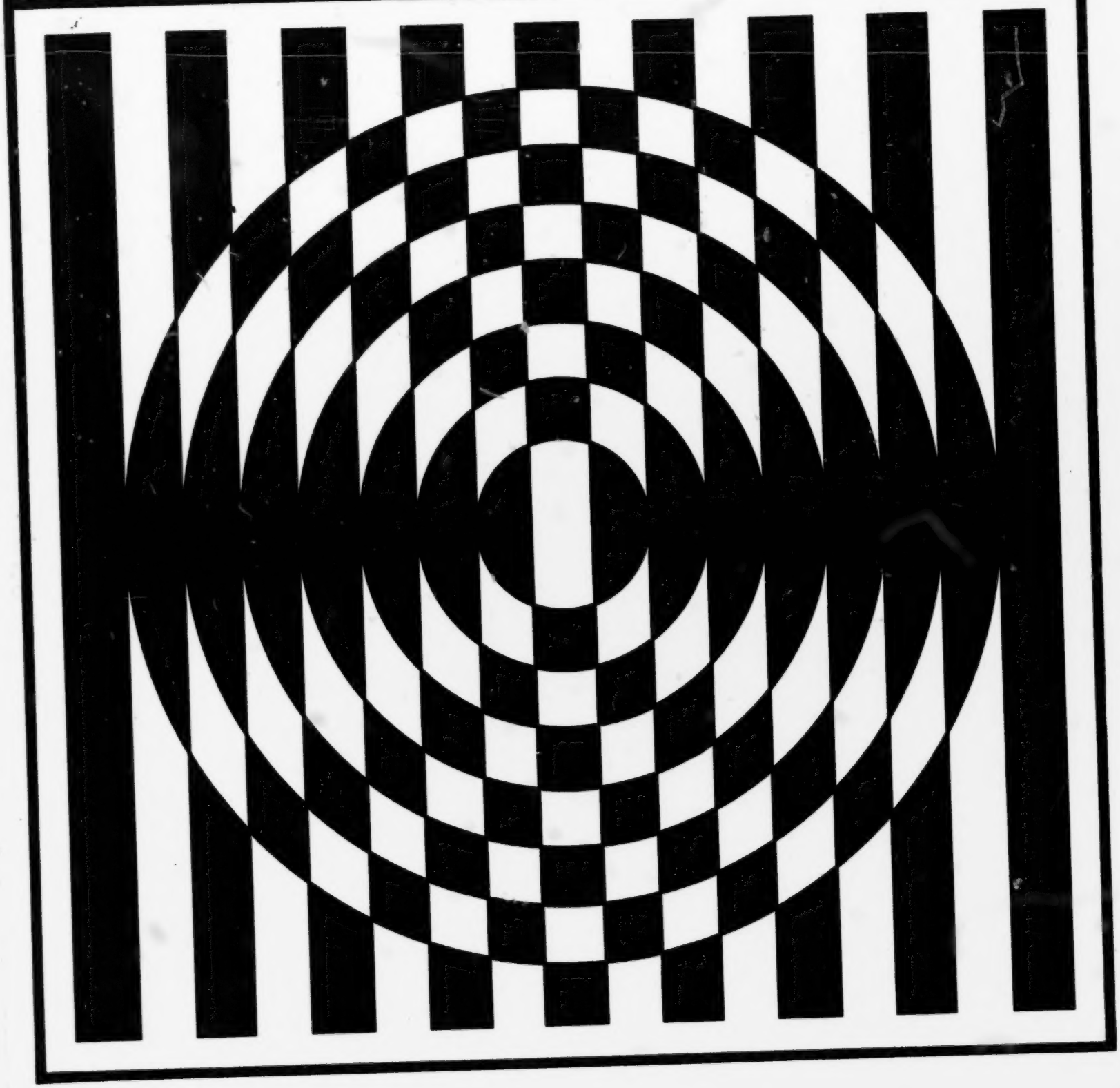
September 1967

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
OFFICE OF EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL OFFICE OF EDUCATION POSITION OR POLICY.

Cosine Committee

Commission on Engineering Education



SE004 120

COMPUTER SCIENCES IN ELECTRICAL ENGINEERING

An Interim Report of the

COSINE COMMITTEE

of the

Commission On Engineering Education
1501 New Hampshire Ave., N.W.
Washington, D.C. 20036

September, 1967

This report has been supported in part by the National Science Foundation under Grant GY-2108.

COMPUTER SCIENCES IN ELECTRICAL ENGINEERING

ABSTRACT: The vital role that electrical engineering departments must play in providing undergraduates with special competence in computer sciences is explored. Three related problem areas are discussed: the needs of students majoring in computer sciences in electrical engineering; bringing into balance the treatment of continuous and discrete systems so that students have a background in discrete systems comparable with that which they now acquire in continuous systems; the means for wider and more effective use of the digital computer as a tool for analysis and design in all engineering courses. Specific suggestions for meeting these needs are made.

* * * * * *

The Committee on Computer Sciences in Electrical Engineering (COSINE Committee) of the Commission on Engineering Education was organized in 1965 to encourage action and suggest direction to electrical engineering departments in meeting the challenge posed by the advent of the computer age.

As part of its activities, COSINE has conducted meetings with representatives of industry, government, and universities on educational needs in computer sciences; has organized summer institutes in computer sciences at Princeton University; and has called meetings of electrical engineering department heads to explore problems relating to computer science programs in electrical engineering.

This report reflects the findings of these meetings and presents the preliminary recommendations of the COSINE Committee for curricular reform in electrical engineering. This report should not be construed as a definitive document, but merely as a set of guidelines for action consistent with the resources available at each institution.

During our study, we have maintained liaison with the ACM Curriculum Committee on Computer Sciences through E. J. McCluskey, and we are especially indebted to William Viavant of that Committee for his assistance. We also wish to acknowledge the support of our efforts by the National Science Foundation through the Commission on Engineering Education, and its executive director, Newman A. Hall.

The COSINE Committee consists of the following:

Jack B. Dennis, Massachusetts Institute of Technology
David C. Evans, University of Utah
William H. Huggins, Johns Hopkins University
Maurice Karnaugh, IBM Federal Systems Division
James F. Kaiser, Bell Telephone Laboratories
Franklin F. Kuo, University of Hawaii
Edward J. McCluskey, Stanford University
Samuel Seely, Educational Consultant (Chairman)
William H. Surber, Princeton University
M. E. Van Valkenburg, Princeton University
Lotfi A. Zadeh, University of California

This report has been prepared under the auspices of the Commission on Engineering Education whose policy is to encourage the exploration of new ideas in engineering education. The Commission has been kept informed of the discussions of the COSINE Committee, but has taken no position on the present report or its recommendations.

I. INTRODUCTION

The rapid growth in the accessibility and power of digital computers for purposes of numerical computation, data processing, storage, control, and retrieval of information, is having a deep, though not necessarily uniform, impact on all branches of science and engineering. While most branches of science and engineering are concerned primarily with the use of digital computers, electrical engineering, by virtue of its long standing and deep involvement in information processing technology, has vital concern not only with the use but more importantly, with the conception, design and construction of digital computers. Furthermore, electrical engineering is deeply involved in a wide gamut of areas which border on or are contributory to computer technology, such as: integrated circuits and microelectronics; switching theory; finite state systems; control systems; communication systems; adaptive and trainable systems; pattern recognition; decision processes; recording, storage and transmission of data; information retrieval, and others.

During the past several years, the rapid growth in the use of computers in science, engineering, business, industry, and many other fields, has tended to shift the emphasis in computer technology from circuit and component design to system organization and programming, or in the roughly equivalent, but more succinct terms, from hardware to software. This trend has given an impetus to the crystallization of what is now widely referred to as computer sciences, that is, an aggregation of subject areas centering on the use of computers as large-scale information processing systems. Chief among these subject areas are: programming systems and languages; machine organization and logical design; formal languages; automata theory; algorithms; heuristic programming; numerical methods; etc.

Clearly, it would be unreasonable to equate computer sciences with electrical engineering, or to regard it as a subset of the latter. Nevertheless, the close relation between the two is presenting the electrical engineering departments with a special responsibility for the training of the large number of computer engineers and scientists who would be needed by industry, government, business, and educational institutions, in the years ahead.

This responsibility derives not only from the close connections between electrical engineering and computer technology, but also from the following: (a) the traditional emphasis in electrical engineering curricula on physical sciences pertinent to information processing; (b) the extensive experience in teaching mathematically oriented subjects relating to signals and systems; (c) the tradition of analyzing system behavior on an abstract level without regard to the physical identity of its variables; (d) the existing

expertise in subjects which fall into computer sciences or are closely related thereto; and (e) the vast resources in facilities and faculty which the electrical engineering departments have at their disposal.

The emergence of computer sciences as a highly important field of study, coupled with the growing shift in emphasis in information processing technology from the analog and the continuous to the digital and the discrete, is creating an urgent need for a major reorganization of electrical engineering curricula. Such a reorganization must, in the first place, accommodate the needs of students who wish to major in computer sciences within electrical engineering. Second, it must bring into balance the treatment of continuous and digital systems, and provide all electrical engineering students with a background in digital systems comparable to that which they currently acquire in continuous systems. Third, it must result in a much wider and more effective use of the digital computer as a tool for system analysis and design in all engineering courses.

How can these objectives be achieved? The COSINE Committee strongly feels that, as an essential first step, electrical engineering curricula should be made substantially more flexible. The movement toward greater flexibility is already under way in most engineering curricula; and it is only in the climate of flexibility that engineering education can respond to the rapid advances in science and technology, and adapt to the explosive growth in knowledge that is now taking place.

The need for flexibility is particularly acute in electrical engineering not only because it is one of the fastest changing fields in natural sciences but also because of the wide diversity of subject areas within electrical engineering. Apart from computer sciences, the fields of solid state electronics, quantum and optical electronics, microelectronics and integrated circuits, bioelectronics, plasmas, circuits, systems, control, communication systems, large-scale power systems, are but the more prominent of the many important subject areas which comprise electrical engineering. Each of these areas has its own needs and objectives which, in many cases, cannot be satisfactorily met within the framework of a single curriculum with just a few electives and a large core of required courses in engineering and electrical engineering.

Thus, a climate of flexibility is essential for the accommodation of the needs of computer sciences as well as other subject areas within electrical engineering. It should be noted that some electrical engineering curricula already offer the student close to a full year of electives with more available by route of petition. In most cases, this is sufficient to enable a student to focus his studies in an area of concentration which may be computer sciences or computer sciences in combination with such areas as circuits, systems, control, or solid state devices. The Committee feels that flexibility of this order of magnitude would permit many diverse educational programs to thrive within electrical engineering and make it possible for students in such programs to acquire excellent training in both the foundation subjects in electrical engineering and the more specialized subjects in their particular fields of interest. Indeed, such training would serve well not only the needs of students who would continue their studies toward higher degrees, but also those who would terminate their formal education at the undergraduate level.

The major recommendations made in this report are in three parts. These are detailed under the headings, II. A Computer Science Program in Electrical Engineering; III. The Implication of the Digitalization of Information Processing Technology; IV. The Impact of Computers on Courses in Circuits, Systems, and Related Areas. As the title implies, Part II is concerned, in the main, with education in the principles and practice of the design of general purpose digital information processing systems. Part III focuses attention on the shift of information processing technology from the continuous to the discrete, a shift which demands radical changes in the conceptual framework of electrical engineering education. Part IV is concerned with the integration of digital computation into many of the traditional courses in the electrical engineering curriculum. A number of specific suggestions concerning courses in the systems area, such as circuits, linear systems and control systems, are discussed. These show the manner and establish a pattern of change that might be used in other courses in the electrical engineering program.

II. A COMPUTER SCIENCE PROGRAM IN ELECTRICAL ENGINEERING

By a computer science program in electrical engineering, the Committee means a curriculum in electrical engineering education which allows the student to acquire substantive competence in computer sciences and related fields, comparable, but not necessarily similar in content, to that acquired by students in a typical computer science department. Such a program must fulfill the following aims:

a. It must provide the student with a thorough understanding of computer systems and their use that is based on fundamental principles of long term value rather than the salient facts of contemporary practice.

b. It must give the student a background in the relevant discrete mathematics (set theory, mathematical logic, and algebra) including familiarity with methods of deduction as applied to abstract models relevant to the field of computation.

c. It must give the student access to a variety of subjects covering specialized and advanced aspects of computer science.

d. It must provide the student with sufficient technical and general knowledge so that he can readily broaden his education through continuing study, and remain adaptable to the changing demands of society throughout his professional life.

Our discussion relates to the first three of these objectives; achieving the last objective is left to the discretion of each university.

The Committee recognizes the inherent difficulty in attempting to specify a detailed curriculum in computer science; no single curriculum could possibly fit into the variety of programs and organizational frameworks present in electrical engineering departments. We have, therefore, organized the material into subject areas as shown in Table II-1. Each subject area is a collection of related topics having certain cohesion and purpose. In describing a subject area, the Committee does not wish to imply that it necessarily corresponds to a single one-semester course. For example, subject area A-4: Machines, Languages and Algorithms, might be covered in a one semester course, or possibly in a two semester or two quarter sequence. Furthermore, the description provided for each subject area is intended only to indicate what the Committee regards as a reasonable set of topics and their logical order, without implying that strict adherence to the description is to be expected.

TABLE II-1. Subject Areas for a Computer Science Program in Electrical Engineering

Category A: Basic Subject Areas
A-1. Programming Principles
A-2. Computation Structures
A-3. Introduction to Discrete Mathematics
A-4. Machines, Languages and Algorithms
Category B: Recommended Elective Subject Areas
B-1. Digital Devices and Circuits
B-2. Switching Theory and Logical Design
B-3. Programming Systems
B-4. Operating Systems
B-5. Numerical Methods
B-6. Optimization Techniques
B-7. Circuit and System Theory
B-8. Information Theory and Coding
B-9. Functional Analysis
B-10. Combinatorics and Applications
B-11. Probability and Statistics
B-12. Symbol Manipulation and Heuristic Programming

As seen in Table II-1, the subject areas are divided into two categories. Category A comprises four basic subject areas which, the Committee feels, are of central importance, and are basic to an adequate education in computer science. Other subject areas, which are less central but which, nonetheless, cover important related and specialized material, are listed in Category B. The Committee feels that these subject areas should be available to students in the computer science program. However, we do not view the Category B list as necessarily complete, since there are legitimate differences of opinion about whether certain additional areas should be offered.

A computer science program in electrical engineering can assume a variety of forms. It can start at the freshman, the sophomore, or the junior level. It can be structured as an option with specified required courses, restricted electives, and unrestricted electives. It can be realized by allowing for enough electives in a standard

electrical engineering curriculum to make it possible for a student (with the help of a faculty advisor) to put together a program of his own in computer sciences. It may or may not include a core of required electrical engineering courses in areas outside of computer sciences, e.g., network and system analysis, electronic circuits and devices, electromagnetic theory and its applications, etc. Accordingly, the subject outlines presented below should be regarded as guidelines that are intended to assist electrical engineering departments in devising curricula in computer sciences which are commensurate with their particular needs, circumstances, and available resources.

Finally, the Committee takes no position on jurisdictional questions relating to the departmental responsibility for particular courses. However, since information processing in all of its forms will continue to be of major concern to electrical engineering departments, the Committee feels that electrical engineering faculties should strive to develop strong expertise in computer sciences and related areas. At the same time, it is essential that electrical engineering departments cooperate closely with all departments having interests in computer sciences and share with them the responsibility for providing instruction in computer-oriented courses, and for conducting research in computers and computer-related areas.

II-1. Category A: Basic Subject Areas

We suppose that the student embarking on this program has had a previous exposure to the use of automatic computing, whether in high school or in work experience. In some schools it might be desirable for students in computer science to take immediately the basic course in "Programming Principles." In such cases, some arrangement to provide an early introduction to elementary numerical methods should be provided.

The four subject areas of Category A comprise material that is essential background for all students of computer science. The two subject areas, "Programming Principles" and "Computation Structures" are intended to give students fundamental knowledge of the operation of general purpose computer systems, and the important features of programming languages, with emphasis on computer hardware as the means of realizing programming features. The indicated sequence of development shows our preference for developing familiarity with programming features prior to the discussion of issues of machine organization, instruction, code design, and addressing mechanisms. In this way, it is possible to motivate aspects of machine organization by the language features they serve to implement. This approach also places the conventional machine organization in a less sacred light, and should lead students to consider and evaluate alternative implementations.

The subject area labeled, "Introduction to Discrete Mathematics," is intended to familiarize the student with the mathematical concepts and techniques which are basic to the study of discrete systems. Such familiarity is essential for computer science majors, and certainly very desirable for all electrical engineering students.

The subject area, "Machines, Languages and Algorithms" serves to introduce students to abstract formulations of certain important and related areas of knowledge concerning computation. These areas are not only important in their own right, but they give the student the background and experience that will enable him to make significant use of abstract modeling in his future professional work.

A-1. Programming Principles

This material should be designed to familiarize the student with important concepts in the description of algorithms, the frequently occurring data types and their associated operators, and methods of defining programming languages syntactically and semantically.

A reasonable selection of topics might include the following: Practice in algorithm design and programming to provide familiarity with the primitive operations on commonly encountered data types, e.g., truth values, integers, real numbers, arrays, symbol strings, queues, stacks, trees and lists. Infix and polish notation for expressions and the use of a pushdown list for their intertranslation and evaluation. Assignment operator; conditional expressions; iteration and subscripting. Programs as defining functions with certain domains and ranges. Building complex programs (functions) through the composition (nesting) of more elementary routines: binding of arguments; local and global identifiers and their scopes; sharing; recursion.

A formalism for defining the syntax of programming languages (e.g., Backus-Naur Form (BNF)) should be introduced and the notions of derivation and ambiguity treated. To the extent reasonable, programming constructs should be defined in terms of simpler constructs in order to provide some precision in the treatment of programming language semantics.

Students should acquire a thorough understanding of these concepts through the design of algorithms in which they play important roles, including appropriate use of computing facilities to the extent practicable.

Advanced topics that merit study because of their anticipated importance in future computing systems are: parallelism in the description of algorithms; interprocess communication; and formal systems for defining the semantics of programming languages.

Reference List: See Appendix A-1.

A-2. Computation Structures

The objective of this area of study is to teach the student to appreciate the capabilities and limitations of computer systems technology in realizing the programming features developed in the "Programming Principles" subject. Considerable emphasis

should be placed on the interrelation and trade-offs between hardware and software techniques.

A knowledge of logical design fundamentals is an essential component of this subject. Topics include: the realization of Boolean functions by combinational gate logic, the flip-flop, and registers as ordered sets of flip-flops, register transfer operations, and theory and design of sequential control logic.

Basic topics on number representation and the implementation of arithmetic operations: binary number system; representation of negative numbers; simple mechanizations of addition, multiplication, division; floating point representations of real numbers.

Discussion of memory systems: coordinate addressed and associatively addressed memory structures as abstractions. Practical realizations of memory systems as direct-access memory (magnetic core) and sequential access devices (disk, drum). It would be instructive to treat addressing by key transformation ("hash coding") as a software alternative for associative search.

Discussion of instruction code design is motivated by the process of code generation, i.e., the translation from source language (e.g., the assignment statement) into machine language. Problems of storage assignment, optimization of sequencing, etc. can be discussed. Indexing is introduced as a means of implementing the subscribing operation.

Discussion of the subroutine as a mechanism for the shared use of procedure information at distinct points in a program. The use of indexing and indirect addressing for providing the right context during the execution of shared code. The use of a stack to implement nested (or possibly recursive) routines; stack as a concept in the design of instruction codes.

The implementation of string and/or list operations by software, including a treatment of garbage collection. This serves as an introduction to techniques of dynamic storage allocation.

Discussion of the linking of separately compiled procedures, including the notion of relocatable code and the operation of linking loaders.

Additional topics of an advanced nature that should be considered are: location-independent addressing; multilevel memory organization; multiprocessing.

Reference List: See Appendix A-2.

A-3. Introduction to Discrete Mathematics

This subject area is intended to provide a foundation of knowledge upon which the student may undertake advanced subjects involving applications of discrete mathematics

(such as subject area A-4), or deeper study of discrete mathematics itself (e.g., modern algebra, mathematical logic, etc.). Central to the student's ability in this area is an understanding of axiom systems and methods of rigorous proof. It is suggested that these goals could be accomplished by a study of elementary mathematical logic, formal deduction and proof illustrated by application to important topics in discrete mathematics. A representative set of topics follows: propositional logic; Boolean algebra; set-theoretic notation; axiom systems and formal deduction; formal and informal proofs; proof by contradiction and finite induction; quantification and its use in formalizing propositions; application to the study of formal properties of number systems, graphs, fields, groups and semigroups, linear transformations.

Reference List: See Appendix A-3.

A-4. Machines, Languages and Algorithms

This subject area is intended to cover certain theoretical ideas that are closely related and particularly relevant to a broad knowledge of computation. These ideas fall into three topics:

1. The finite state model
2. Formal languages
3. Computability

An understanding of the finite state model is important in connection with the operation of digital hardware, and is frequently applicable to other processes of interest to the engineer or programmer. The subject, especially when expanded to encompass finite state languages, serves as excellent preparation in thought patterns and mathematical method for the later topics.

Concepts from formal linguistics have had a major impact on work in programming languages. This subject also leads to meaningful and relevant examples of effectively unsolvable problems.

The topic of computability is essential so that students may understand the limitations of mechanical means of computation, and can appreciate the nature of questions that are undecidable by machine.

In teaching this material, more or less emphasis may be placed on the formal methods according to the interest of students and the taste of the instructor. In any case, the principal results should be motivated through the use of non-mathematical arguments so that the concepts are not hidden by the formalism. An important objective is to enhance the student's ability to formulate and work with abstract models of important practical situations.

The following is a representative selection of topics: the finite state model; state diagram and flow table descriptions; equivalent states, equivalent machines; state

reduction; finite state languages, regular expressions and Kleene's theorem; limitations of finite state automata. Formal languages; grammers and derivations; context-free languages and their relation to pushdown storage automata; ambiguity and other properties; sentence parsing procedures. Computability: Turing machines, universal Turing machines; the existence of noncomputable functions; the "busy beaver" and halting problems; unsolvable problems of practical interest, e.g., undecidable properties of context-free languages; the computability of recursive functions; Post systems; Church's thesis.

Reference List: See Appendix A-4.

II-2. Category B: Elective Subject Areas

As already noted, the subjects in this category are less central to a computer science program than those in Category A. Moreover, the list below is not necessarily complete, and individual ideas can be reflected in this group. Short descriptions are given of the subjects to indicate the content and level of the material. The subjects included may denote courses, and in a few cases, they can represent more than one course. The description of other related courses are given in Sec. IV.

B-1. Digital Devices and Circuits

Modeling of nonlinear circuit elements; approximate analysis of quiescent and transient circuit behavior; use of time-domain circuit simulation; designing to specification with imperfect components; worst case and statistical approaches to circuit reliability. Application to flip-flop, multivibrator and networks of cascaded gate circuits; signal transmission methods; integrated circuit technology.

Physical phenomena usable to realize memory function; ferromagnetics, cryogenics, electrostatics, photochromic materials, sound waves; address selection principles — coordinate and serial. High-current switches for inductive loads; sense amplifier design; techniques for improving signal-to-noise performance.

B-2. Switching Theory and Logical Design

Combinatorial logical design, including the notion of prime implicants. Huffman theory of sequential machines, synchronous and asynchronous. Hazards and their resolution. Interconnection of submachines to form larger units. Time-independent logical design. Identification and diagnosing experiments. Error detecting and correcting codes. Languages for specifying digital systems.

B-3. Programming Systems

Formal methods of specifying language syntax and semantics. Syntactic structure, parsing methodology, diagnostics. Advanced study of programming features, e.g., data

structures, properties of data types, block procedures and the context of identifiers, parallelism and sharing of data, protection and process monitoring features. Implementation questions, including symbol table structure, code optimization, efficient subscripting, flow of control analysis and loop organization, efficient subscripting, flow-of-control and loop optimization, subroutine linking and parameter passing, syntax directed compiling.

Comment: See comments on B-4 below.

B-4. Operating Systems

Functions of an operating system: controlling the use of computer system resources by programs submitted for execution by its users; insuring the integrity and security of information held on behalf of users. Topics suitable for in-depth study are: the concept of process, the blocking and awakening of processes, the meaning of interrupts, interprocess communication, and process scheduling. The concept of address space, binding of procedures and data to address space and interprogram linking, motivation for location-independent addressing and techniques of implementation, shared information. Storage management: movement of information within a storage hierarchy, file backup, and issues of data integrity on restart. File access control and transfer of access privilege.

Comment: The material in systems programming divides into two conventionally separate subjects, Operating Systems (B-4) and Programming Language Systems (B-3). However, there is an intimate relationship between the two subjects in that the features implemented by operating systems have a far-reaching and subtle effect on the construction of programming systems. Dynamic storage allocation, inter-process communication, and file organization, are areas of particular concern. These interactions should be emphasized in the organization and teaching of these subjects. Perhaps they should not be offered as parallel subjects, but as a sequence in which both aspects of each topic are studied together.

B-5. Numerical Methods

Solution of systems of linear equations (matrix inversion, Gauss elimination, determinants, etc.); numerical solution of nonlinear algebraic equations, roots of polynomials, interpolation techniques and curve fitting, numerical integration, solution of ordinary differential equations, solution of partial differential equations, linear programming.

Comment: This course would ordinarily be taken in the senior or junior year. Unlike conventional courses in numerical analysis in which computation plays a minor role, this course should be strongly oriented toward the use of computers, and should stress computational efficiency of various algorithms, the effect of round-off error and truncation error, convergence of approximations, etc.

B-6. Optimization Techniques

Solution of linear inequalities, linear programming algorithms, convex sets and convex functions, nonlinear programming, quadratic programming, dynamic programming, gradient techniques, maximum principle, Markoffian decision process, optimization under vector-valued criteria, search strategies.

Comment: This material would normally be covered in a two-semester course. The subject matter is primarily of interest to those computer science majors whose minor is in systems, control, or operations research.

B-7. Circuit and System Theory

Circuits as interconnections of basic elements: passive and active circuits, characterization of circuits in the time and frequency domains, solutions of differential input-output relations. State-space formulation: representation by differential and difference equations. Basic properties of linear systems, time-varying systems and nonlinear systems. Controllability, observability and stability.

Comment: This is not expected to be a conventional course in circuit analysis. Rather, it is supposed to be a more general type of course in which fairly heavy emphasis is placed on basic concepts in system theory, with particular consideration of the techniques and applications of state-space formulations. To cover the material in adequate depth, a two-semester sequence would probably be necessary.

B-8. Information Theory and Coding

Quantitative definition and measurement of information, entropy and uncertainty, memoryless discrete channel, capacity of a memoryless channel, capacity theorems. Encoding and decoding of messages, parity check codes, convolutional encoders and decoders, sequential coding. Practical digital communication systems.

Comment: A familiarity with the concepts and techniques of information theory should constitute an important part of the basic training of any student who intends to pursue a career in information processing and related areas. The emphasis in this course is on discrete signals and channels, for which the student need not possess an extensive background in probability and statistics.

B-9. Functional Analysis

Functions, functionals and operators. Metric and topological spaces, linear spaces, Hilbert spaces. Linear functionals, differentiation of abstract functions, homogeneous forms and polynomials. Stationary problems, fixed point theorems, gradient techniques. Quasi-linearization. Applications to problems in optimization and identification.

Comment: This course is intended to provide the student with basic mathematical techniques for dealing with problems in numerical analysis and optimization. It is envisaged as a broad course, not necessarily oriented specifically toward the needs of computer science majors.

B-10. Combinatorics and Applications

Enumeration techniques, including permutations and combinations, generating functions, recurrence relations, the principle of inclusion and exclusion, Polya's theory of counting. Theory of graphs, including planar graphs and duality. Network flow problems and elementary linear programming.

B-11. Probability and Statistics

The concept of sample space and random variable, probability distributions on discrete sample spaces, dependent and independent random variables, conditional distributions, distributions on continuous sample spaces, parameters of probability distributions, normal distributions, stochastic processes, Markoff chains, waiting line and servicing problems, estimation techniques, stochastic approximations, decision rules.

Comment: A basic course in probability and statistics should be strongly recommended to all students majoring in computer sciences. The course should include a discussion of time series and Markoff processes, since these topics are of particular relevance to problems involving the transmission of information, stochastic service systems, etc.

B-12. Symbol Manipulation and Heuristic Programming

Heuristic versus algorithmic methods; LISP and other relevant programming methods, game playing programs, question-answer programs, symbolic integration and differentiation, theorem proving, search techniques, simulation of learning and concept formation, applications to pattern recognition and information retrieval.

Comment: Although heuristic programming is not as yet a well-developed subject area, it has considerable potential importance in fields such as game playing, pattern recognition, information retrieval, etc. in which decision making must be based, for the most part, on heuristic procedures. The primary purpose of this course is to introduce the student to computer-oriented heuristic problem solving techniques.

II-3. Computer Sciences in an Electrical Engineering Curriculum

To place the foregoing discussion into the context of a four-year undergraduate program, refer to Table II-2. This table contains only the basic courses in Category A

plus an introductory course in Programming and Numerical Methods. Appendix B of this report contains the curricula of several electrical engineering departments, which show how they have included a concentration in computer sciences.

TABLE II-2. Skeleton of Program Showing Recommended Courses in Computer Science

Year	Term 1	Term 2
Freshman	---	Programming and Numerical Methods*
Sophomore	---	---
Junior	Programming Principles**	Computation Structures**
Senior	Machines, Languages and Algorithms**	---

*The programming course that is normally available at most universities.

**The arrangement of individual topics to form a course or courses will depend on local conditions.

NOTE: The material of course A-3, Introduction to Discrete Mathematics, may be introduced into the curriculum in various ways depending on the local circumstances.

III. THE IMPLICATIONS OF THE DIGITALIZATION OF INFORMATION PROCESSING TECHNOLOGY

During the past two decades, as a result of the invention and development of a number of electronic components, circuits and devices, such as the transistor, the magnetic core memory, integrated circuits, etc., it has become practicable and economical to process large volumes of data in digital form with high speed, accuracy, and reliability. We have witnessed a rapidly growing trend toward the use of digital systems in place of analog or continuous systems for purposes of computation, information processing and control. Moreover, as a result of the availability of efficient, economical and reliable digital devices, modern information processing and control technology is becoming increasingly digital in nature, with all signs pointing toward a much bigger role for digital as compared with analog systems in the years ahead.

The transition from the analog and the continuous to the digital and the discrete has not yet been adequately reflected in the orientation of electrical engineering curricula. Many, and perhaps most, electrical engineering departments still lay a heavy stress in their curricula on courses in continuous (in time, amplitude, and state) systems and devices, disregarding the fact that such courses are much less relevant to the needs of present technology, and certainly much less relevant to the needs of the future, than they were twenty years ago, in the age of the vacuum tube and the amplidyne. The Committee strongly feels that, in this regard, electrical engineering curricula are in need of a basic reorientation from the entirely analog and the continuous to reflect the digital and the discrete, and that electrical engineering departments should make a concentrated effort to prepare their students to deal with digital systems, be they computers, control systems, or special purpose information and data processing systems.

How can such a reorientation be implemented? Clearly, a wide-ranging shift in emphasis from the continuous to the discrete in electrical engineering curricula would present formidable problems which are not likely to be solved quickly or painlessly. Deeply entrenched attitudes will have to be changed, new knowledge and skills will have to be acquired, and new textbooks will have to be written. Indeed, it is beyond the scope of this report to analyze these problems fully, and to suggest possible solution to them. Thus, in what follows, the Committee will restrict itself to making a few preliminary recommendations that suggest evolutionary changes in electrical engineering curricula. This section will discuss the addition of three new courses dealing wholly or in part with some of the basic aspects of discrete systems. Section IV will discuss the digital reorientation of a number of existing courses.

III-1. New Course Development

To provide a start toward the development of new courses that have a discrete state orientation, we discuss three possible courses as indicative of the direction that such development might take.

Course 1

Our first recommendation is that serious consideration be given to the development of a sophomore or junior level introductory course in circuits, systems and signals, which would cover the fundamentals of both discrete and continuous type systems. We envisage that, initially, such a course would be offered as an alternative to the traditional type of course in which the emphasis is wholly on the techniques of time- and frequency-domain analyses of linear, time-invariant, continuous-time circuits and systems. Eventually, courses of this new type would probably replace introductory courses of the more conventional nature.

An example of the type of course being recommended is that being developed at MIT by Professors Athans, Dertouzos and Mason, under the title, ELEMENTS, SYSTEMS and COMPUTATION. In addition to covering the basic techniques of the analysis of linear, time-invariant, lumped-parameter networks and systems, this course also treats basic techniques for the study of nonlinear and discrete-state systems, and covers computational as well as analytic methods of problem solving in the context of such systems.

A controversial aspect of a course of this type is that its broader coverage of both continuous and discrete systems is attained necessarily at the cost of less depth in the treatment of various types of components and systems. For this reason, it may be preferable, in the longer run, to treat discrete systems separately in a course that would precede a course in continuous systems. Although this would represent a departure from the traditional order, it may well be more logical and more sound pedagogically.

Course 2

Our second recommendation relates to the inclusion of a new type of course at the junior or senior level which would be concerned, in the main, with mathematical concepts and techniques which are central to the analysis and synthesis of discrete, as contrasted with continuous, systems. The importance of such a course was discussed in Section II-2, and is course A-3 of the Category A group.

A representative set of subjects that might be included in a course of this type would be: elements of set theory, Boolean algebra, elements of mathematical logic, elements of the theory of relations, groups, fields and rings, elements of Galois theory,

etc. A course of this type would serve essentially the same function in relation to discrete systems that the conventional courses in Laplace transforms, complex variables, linear algebra, etc. serve in relation to the analysis of linear time-invariant systems. Clearly, of course, the totality of the mathematical background needed for the analysis and synthesis of discrete systems cannot be provided in a single course.

A theoretically oriented student majoring in computer sciences might well take one or more courses in mathematics, in such subjects as: abstract algebra, set theory, mathematical logic, group theory, etc., in preference to taking a single less specialized course of the type here being discussed. Thus, the Committee's recommendation is intended primarily to point to a need in electrical engineering curricula for a broadly based course in the mathematics of discrete systems which would be suitable for most electrical engineering students, and not just for those majoring in computer science. A desirable first step in this direction may be accomplished by a revision of the usual two-year mathematics program that exists in almost all electrical engineering curricula (the Calculus program), to a program, one half of which is devoted largely to discrete mathematics, the second half being devoted to topics in continuous mathematics.

Course 3

Our third recommendation relates to the offering of a course in finite-state systems at the junior or senior level. The importance of such a course stems from the fact that finite-state systems constitute a very basic class of systems particularly well-suited for the introduction of such basic concepts as state, equivalence, identification, decomposition, etc. Furthermore, they are much better suited for computational purposes than continuous systems, and can frequently be used as approximate models for the latter. At present, several electrical engineering departments offer courses of this type, covering such topics as: the characterization of finite-state systems, the notions of state and system equivalence, identification algorithms, decomposition techniques, synthesis techniques, etc.

Until a few years ago, the offering of courses on finite-state systems was hampered by the dearth of texts on this subject, a situation that is now changing. There are several very good undergraduate level texts on finite-state systems. The teaching of a course on this subject should be a relatively easy task for most electrical engineering professors. Consequently, the Committee feels that every electrical engineering department should consider offering an elective course on finite-state systems as part of its regular curriculum.

IV. IMPACT OF COMPUTERS ON COURSES IN CIRCUITS, SYSTEMS AND RELATED AREAS

As has already been noted, the digital computer has brought about profound changes in engineering analysis and design techniques. It is strongly recommended that electrical engineering curricula should include increased emphasis on computer methods, wherever this is appropriate, to provide a more realistic preparation for the practice of engineering. The problem is more than that of merely adding material to traditional courses, since many traditional approaches to the solution of system equations are not particularly convenient for digital simulation. For example, the state-variable formulation of systems, which has had increasing acceptance, is especially well adapted to computer study and should be emphasized at the expense of other formulations. In addition, the use of approximate analytical methods for studying nonlinear systems is often far more cumbersome than adopting computer simulation techniques which can provide numerical solutions.

The implementation of the necessary changes in undergraduate electrical engineering programs has been relatively slow, due to many factors. Since suitable textbooks are not yet generally available, a thorough revision of an electrical engineering course in a traditional area would involve a major effort by the instructor. As a result, the need for a computer science viewpoint is often met by merely grafting a few computer examples on top of a conventional presentation. Another factor which has retarded the development of computer-oriented courses is the lack of suitable "educational software," which would provide easy problem-oriented input instructions and graphic output.

This section will be concerned with the changes that can be made in course presentations to take advantage of the existence of large computing systems, and to acknowledge the greatly increased importance of digital techniques of information processing.

IV-1. Implementation

To introduce computer techniques in a meaningful way into traditional courses covering circuit theory, control systems, communication systems, and similar topics, the following sequence might be used in presenting subject matter: theory, analytical methods of solution, numerical algorithms, and computational examples. Some general

recommendations are outlined below, and a set of more specific examples of course revisions are given in Section IV-2. These recommendations presuppose a familiarity with programming principles and elementary numerical methods.

The following may be stated as general objectives:

1. The first course in electrical engineering (usually in the sophomore year) should be modified to incorporate the use of computers as a tool. A problem-oriented program having a special language might be used to work exercises relatively early in the course, without requiring a detailed knowledge of computer programming.

2. Major revisions might be made in the method of presentation for certain courses, particularly those in the systems area. For programs in control theory, and communication systems, for example, this might involve changes in emphasis of some of the traditional material and introducing new material related to computer operation and limitations.

3. Since the purpose of courses in the systems area is to develop an understanding of the behavior of these systems, it would be desirable to make available to the students as analysis and design aids certain fairly elaborate specialized computer programs with provision for graphic output. This would provide computer-generated results without the large investment of the student's time that would be required were he forced to write and debug all of the necessary programs.

4. The academic program should help develop a more thorough understanding of programming techniques and the limitations of numerical methods of simulation, particularly in those areas in which the computer is a major factor in the practice of modern engineering, either as a design tool or as an important part of the system.

5. The use of computers for system simulation should be encouraged as a valuable supplement to laboratory experiments with physical elements.

IV-2. Examples of Course Revisions

Clearly, there are a large variety of ways in which an electrical engineering curriculum can be organized. Similarly, the content of an individual course will depend on the sequence of courses in which it is embedded. For example, a course which treats the state-variable formulation of the system differential equations with analytic and computational methods of solution requires a certain background in linear algebra. Whether or not it is necessary to include details of certain aspects of linear algebra in the engineering courses will depend on the mathematical background of the students.

Each course description given below is intended to illustrate one possible method of incorporating a computer-oriented approach within one of the standard type courses usually available in electrical engineering curricula.

1. INTRODUCTORY COURSE IN CIRCUIT THEORY (Sophomore Level). The usual introductory course in circuit theory deals with methods for formulating circuit equations, and uses analytical methods to describe the dynamic circuit behavior, with little or no use of digital computation. A suggested sequence is the following: First, some basic concepts in network topology should be presented. Within the presentation are included certain algorithms dealing with connectivity, determining the shortest path between two nodes, etc. Next, Kirchhoff's laws are discussed, and the procedures for writing branch, mesh, and node equations for d-c networks are given. Once formulated, these network equations constitute a linear set of algebraic equations. Numerical algorithms, such as the Gauss-Jordan and Crout methods are introduced to solve the network equations. Next, steady-state analysis is presented. The numerical algorithm relevant to this material involves extension of the linear equation solving techniques to the case of equations with complex coefficients. Computational techniques for magnitude and phase response evaluation, including Bode plots, should also be included. Then the transient analysis, using ordinary differential equations should be discussed. Runge-Kutta or equivalent methods are presented in conjunction with the theoretical discussions. State-space formulation for transient analysis are then given, along with computational methods for eigenvalue and eigenvector evaluation. Next, various signal representation schemes are discussed. Computer programs used in conjunction with this section deal with convolution, correlation, Fourier series, and fast Fourier transforms. Finally, the basic concepts of power and energy are discussed, along with techniques for numerical integration.

2. ENGINEERING ANALYSIS AND COMPUTATION METHODS (Junior or Senior Level). The objectives of this course are to provide a study of continuous-time linear and nonlinear systems, together with a systematic development of numerical methods and analytical methods for the analysis of the behavior of such systems. The state-variable method of formulating the system equations is stressed, with emphasis on the effects of nonlinearities on the behavior of both linear and nonlinear systems. The analytical methods of solution are compared with techniques based on digital and analog simulation. Numerical methods are analyzed in sufficient detail to illustrate clearly their limitations, and thereby provide a sound basis for the use of digital computers in the study of large systems.

A brief topical description would be: linear and nonlinear models of physical devices and systems; state-variable formulation of system equations, state space solution trajectories. Methods of solution for linear systems (brief development, partially a review); time and frequency domains, convolution, Fourier and Laplace transforms, poles and zeros, stability. Introduction to the analysis of nonlinear systems; phase-plane methods, Liapunov functions and stability, limit cycle oscillations, describing functions for periodic excitation. Numerical methods of analysis and the digital simulation of dynamic systems; sampling and interpolation, numerical integration, one-step (Runge-Kutta) and multi-step (predictor-corrector) methods of solving differential equations, difference equations and the stability of numerical methods, error analysis. Comparison of digital and analog simulation techniques.

3. LINEAR SYSTEMS THEORY (Senior Level). The traditional development of linear systems theory for continuous time systems should be modified to include the parallel development of the linear theory of discrete-time systems. This will better reflect the rapidly increasing use of digital systems, perhaps including real time computers, and mixtures of continuous-time and discrete-time components with appropriate interfaces.

4. CONTROL SYSTEMS THEORY (Senior Level). The study of control systems provides an ideal opportunity to make use of the techniques of digital simulation. Computers are widely used in practice as a design aid in developing control systems, and are also frequently incorporated as real-time components of large high-performance controllers. It would be highly desirable, as well as realistic, in the first course in this area, to stress the use of the computer in exercises as a tool to assist in the analysis and synthesis of controllers.

Control system theory is a highly developed field, and this discussion will focus on computer applications. The emphasis should be on the study of the behavior of feedback control systems, using the computer to supplement the analytical tools. To do this, however, requires the availability of adequate software. For example, a computer routine to generate root-locus plots can be used very advantageously by students to develop an appreciation of how a variation in a parameter of a linear system affects its overall behavior. This routine should be available in the computer library and should be provided with a graphical output.

At this stage of their academic program, students should be sufficiently sophisticated to understand the operation of computer numerical analysis routines, including the type of errors to expect, and how the system equations should be formulated to minimize them. But they should not be required to divert time from the subject matter of the course to write and debug elaborate computer programs. The computer must be a help, and not a hindrance!

The availability of an adequate computer system (including problem-oriented software) can be extremely useful by allowing significant results to be obtained without the laborious computations otherwise required. It also allows the study of nonlinear systems to be carried out quantitatively almost as easily as for linear systems, it avoids the restriction of artificial indices of performance, and it easily includes the study of the effects of parameter tolerances and "worst case" analyses. The behavior of the system for realistic types of input signals can also be included, i.e., random signals, rather than step or impulse inputs.

IV-3. Computer Use for Simulated Laboratory Experiments

Digital computer simulation of devices and systems can provide a very valuable supplement to laboratory experiments with physical elements. Simulation studies, provided that adequate software is available, can also be used very effectively as

alternatives to some problem sessions and homework exercises. It is recommended that such "computer experiments" on idealized models of physical devices be introduced into the laboratory program early in the curriculum. The essential features of the behavior of many types of systems can sometimes be explored more readily in this way than by actual experiments.

Idealized models can, of course, be nonlinear and can provide quite realistic representations of the true device characteristics. Some advantages of a computer study are the ease with which the model parameters can be varied over a wide range without damage to the components, the ability to compute sensitivity coefficients and make a "worst case" analysis, and the ability to generate and plot performance curves directly for nonlinear as well as linear systems without tedious experimental tests or the drudgery of repeated hand computations. Furthermore, this approach allows a considerable degree of individual initiative to be exercised by the student in the design of the system model to be simulated.

Observation of the behavior of the actual physical system is very important, of course, and experiments on real devices and systems should clearly be retained. Both types of experiments are significant in different ways in helping to develop an intuitive feeling for system behavior; a balance should be maintained between actual and simulated experiments. A combination of both will be more stimulating than either type alone, and has the added advantage of providing a basis for appreciating the differences between the analysis of an "idealized model" and the behavior of the real device. As a specific example, consider the electronics laboratory described below.

ELECTRONICS LABORATORY. The digital simulation of models of electronic circuits can be a significant adjunct to the normal electronics laboratory experiments. A number of quite elaborate electronic circuits analysis computer programs are available, or are being developed. Although many of these have serious deficiencies from an educational viewpoint, there is some expectation that this will be rectified in the relatively near future. An electronic circuit simulator, such as ECAP, when supplemented by a graphical output routine, can provide the basis for some very useful simulation studies in this field.

Examples might include the study of transistor amplifier operating point stability with respect to parameter variations, the transient and frequency response characteristics of pulse amplifiers, and oscillation phenomena.

IV-4. "Educational" Software

The type of software that is necessary to make a digital computer system into a really significant tool for undergraduates is similar, in many respects, to that developed for industrial use in computer-aided design studies. Special subroutines and problem-oriented programming systems, with relatively simple input instructions and with graphic output, should be available. Parameter modifications should be readily

accomplished. Complete parameter optimization routines, while very valuable industrially, are somewhat less effective educationally. We note that the type of output which generates masses of data in tabular form provides a very poor match to the human brain. Even a fairly rough printer plot routine would be quite satisfactory in many cases.

Frequent student-machine interaction would be very desirable, an on-line access to the computer with an immediate graphic display output being the most desirable arrangement. This is not yet generally available. However, a fast turn-around time should be provided for short jobs.

The most desirable types of specialized software that should be available in the computer library can be divided into several general categories, as follows:

1. Mathematical subroutines. Examples would be graphic output subroutines, programs to solve simultaneous sets of linear equations, to find the roots of polynomials, and to solve sets of ordinary differential equations. These subroutines are presently available at almost all computer centers. They are far from being sufficient, however.

2. More elaborate programs to facilitate special types of analysis. Examples would be: frequency response, root locus and transient solution plotting, Fourier series analysis routines, parameter optimization routines, and some statistical analysis routines.

3. Problem oriented programs with special language facilities. Examples of such programs are: analog system simulators, such as MIMIC and CSMP; digital system simulators, such as BLODI; and electronic circuit simulators, such as ECAP. Another significant example is JOBSHOP, which was developed by W. H. Huggins at The Johns Hopkins University, as a simulator for the circuit design process. Much remains to be done in the development of suitable software for educational purposes. In addition, information concerning the availability of the computer programs which do exist, and their documentation, leaves a great deal to be desired.

V. CLOSING COMMENTS

The recommendations presented in this report relate to what the Committee believes are the central issues in the impact of computers and computer sciences on electrical engineering education. These are: (a) the need for computer science programs in electrical engineering, (b) the need for greater emphasis on discrete systems in electrical engineering curricula, and (c) the need for modifying the content and underlying philosophy of basic electrical engineering courses, particularly in circuits and systems, to interweave the use of computers for analysis and design with the development of basic theory.

These issues are probably the most pressing of the many questions and problems facing electrical engineering education today. However, they are by no means the only issues arising out of the advent of the computer age. Clearly, the use of computers will have to be stressed not only in courses in circuits and systems but, more generally, in all areas in electrical engineering in which computers can be an effective tool for analysis, design, or simulation. We have said nothing concerning the roughly three years of studies which constitute that portion of the B.S. program that reflects the general base of electrical engineering. However, we do stress that attention must be given to the revision of introductory courses in mathematics, physics, and other basic fields, with a view to increasing the emphasis on algorithmic and numerical techniques in such courses. Also, the traditional role of laboratory courses must be reexamined, in the light of the possibility of using computers as simulators of physical systems. Already, in many instances, greater insight into system behavior may be obtained by studying its performance with the aid of a computer than by measuring the physical variables and parameters associated with it. Clearly, this will be even more true in the future.

The Committee plans to consider these and other issues relating to the impact of computers on electrical engineering education in its later reports. In studying these issues, the Committee will seek advice, and shall consult with other groups which are studying the impact of computers in their fields of interest, especially the Curriculum Committee on Computer Science of the ACM,[1] and the Committee on the Undergraduate Program in Mathematics of the MAA.[2] It hardly needs saying that computers and computer sciences are, and will be, of considerable concern to many disciplines in addition to electrical engineering, and electrical engineering departments will have to cooperate closely with other academic departments, especially computer sciences and mathematics departments, both in instruction and in research in computers and related areas.

[1] Association for Computing Machinery.

[2] Mathematical Association of America.

APPENDIX A

Selected References for Basic Subject Areas

A-1. Programming Principles

1. B. W. Arden, An Introduction to Digital Computing, Addison-Wesley, 1963, Chapters 1, 2, 3, 4, 11, 17, 18.

2. P. M. Sherman, Programming and Coding Digital Computers, John Wiley and Sons, 1963.

3. Peter Naur, Ed., Revised report on the algorithmic language ALGOL 60. Comm. of the ACM, Vol. 6, No. 1, January 1963, pp 1-17.

4. H. Bottenbruch, Structure and use of ALGOL 60. Journal of the ACM, Vol. 9, April 1962, pp 161-221.

5. E. W. Dijkstra, A Primer of ALGOL 60 Programming, Academic Press, 1962, (includes reference 3)

6. D. J. Farber, R. E. Griswold and I. P. Polonsky, The SNOBOL 3 programming language. Bell System Technical Journal, Vol. 45, No. 6, July-August 1966, pp 895-944.

7. John McCarthy, Recursive functions of symbolic expressions and their computation by machine. Part I. Comm. of the ACM, Vol. 3, No. 4, April 1960, pp 184-195.

8. J. A. Foster, List Processing, MacDonald, 1967.

9. IBM Corporation, IBM System 1360 Operating System, PL/1: Language Specifications. Document C28-6571-4.

10. George Radin and H. P. Rogoway, NPL: Highlights of a new programming language. Comm. of the ACM, Vol. 8, No. 1, January 1965, pp 9-17.

11. K. E. Iverson, A Programming Language, John Wiley and Sons, 1962.

12. J. K. Iliffe, The use of the GENIE system in numerical calculation. Annual Review in Auto. Prog. Vol. 2, Pergamon, London, 1961, pp. 1-28.

13. P. J. Landin, A correspondence between ALGOL 60 and Church's lambda-notation: Part 1. Comm. of the ACM, Vol. 8, No. 2, February 1965, pp 89-101; Part 2. Comm. of the ACM, Vol. 8, No. 3, March 1965, pp 158-165.

14. W. H. Burge, A reprogramming machine. Comm. of the ACM, Vol. 9, No. 2, February 1966, pp 60-66.

15. E. W. Dijkstra, Cooperating Sequential Processes, Technological University, Eindhoven, The Netherlands, 1965.

Books 1 and 2 are good introductory treatments of programming. References 3, 6, 7, 9, and 11 are descriptions of important programming languages. References 4, 5, 6, 7, 8, 10, 11, 12 include useful discussions of semantic features of programming languages. Papers 7, 13 and 14 are approaches to a formal treatment of program semantics. Reference 15 is a readable and neatly integrated discussion of parallel processing.

A-2. Computation Structures

1. E. J. McCluskey, Introduction to the Theory of Switching Circuits, McGraw-Hill, 1965.

2. Y. Chu, Digital Computer Design Fundamentals, McGraw-Hill, 1962.

3. G. A. Maley and J. Earle, The Logic Design of Transistor Digital Computers. Prentice-Hall, 1963.

4. Ivan Flores, The Logic of Computer Arithmetic, Prentice-Hall, 1963.

5. A. D. Falkoff, Algorithms for parallel-search memories. Journal of the ACM, Vol. 9, No. 4, October 1962, pp 488-511.

6. W. W. Petersen, Addressing for random-access storage. IBM Journal of Research and Development, Vol. 1, No. 2, April 1957, pp 130-146.

7. C. L. Hamblin, Translation to and from polish notation. The Computer Journal, October 1962, pp 210-213.

8. C. B. Carlson, The mechanization of a push-down stack. AFIPS Conference Proceedings, Vol. 24, Spartan Books, 1963, pp 243-250.

9. R. W. Floyd, An algorithm for coding efficient arithmetic operations. Comm. of the ACM, Vol. 4, No. 1, January 1961, pp 42-51.

10. B. W. Arden, B. A. Galler, and R. M. Graham, An algorithm for translating Boolean expressions. Journal of the ACM, Vol. 9, No. 2, April 1962, pp 222-239.

11. Thomas Marill, Computational chains and the simplification of computer programs. IEEE Transactions, Vol. EC-11, No. 2, April 1962, pp 173-180.

12. T. C. Chen, The overlap design of the IBM System/360 Model 92 central processing unit. AFIPS Conference Proceedings, Vol. 26, Part II, 1964, pp 73-80.

13. J. F. Thorlin, Code generation for PIE (parallel instruction execution) computers. AFIPS Conference Proceedings, Vol. 30, Thompson Books, 1967, pp 641-643.

14. L. P. Horwitz, R. M. Karp, R. E. Miller and S. Winograd, Index register allocation. Journal of the ACM, Vol. 13, No. 1, January 1966, pp 43-61.

15. Kirk Sattley, Allocation of storage for arrays in ALGOL 60. Comm. of the ACM, Vol. 4, No. 1, January 1961, pp 60-65.

16. E. W. Dijkstra, Recursive programming. Numerische Mathematik, Vol. 2, 1960, pp 312-318.

17. B. Randell and L. J. Russell, ALGOL 60 Implementation, Academic Press, 1964.

18. T. Kilburn, D. B. G. Edwards, M. J. Lanigan, and F. H. Sumner, One-level storage system. IEEE Transactions, Vol. EC-11, No. 2, April 1962, pp 223-235.

19. J. McCarthy, F. J. Corbato, and M. M. Daggett, The linking segment subprogram language and linking loader. Comm. of the ACM, Vol. 6, No. 7, July 1963, pp 391-395.

20. J. K. Iliffe and Jane G. Jodeit, A dynamic storage allocation scheme. The Computer Journal, Vol. 5, October 1962, pp 200-209.

21. J. B. Dennis, Segmentation and the design of multiprogrammed computer systems. Journal of the ACM, Vol. 12, No. 4, October 1965, pp 589-602.

22. B. W. Arden, B. A. Galler, T. C. O'Brien and F. H. Westervelt, Programming and addressing structure in a time-sharing environment. Journal of the ACM, Vol. 13, No. 1, January 1966, pp 1-16.

References 1 through 4 cover material on logical design and arithmetic operations. Partial material for a discussion of memory systems is found in 5 and 6. Material relating to compiling techniques and code generation is contained in 7 through 17. Papers 18 through 22 are concerned with program linking, location-independent addressing and memory hierarchies.

A-3. Introduction to Discrete Mathematics

1. Paul R. Halmos, Naive Set Theory, D. Van Nostrand, 1960. Chapters 1 through 10.

2. Elliot Mendelson, Introduction to Mathematical Logic, D. Van Nostrand, 1964.

3. S. C. Kleene, Mathematical Logic, John Wiley and Sons, 1967. Chapters 1 through 4.

4. Robert R. Stoll, Set Theory and Logic, W. H. Freeman, 1963. Chapters 1 through 8.

5. Hartley Rogers, Jr., An example in mathematical logic. Amer. Math. Monthly, Vol. 70, No. 9, November 1963, pp 929-945.

6. Claude Berge, Theory of Graphs and Its Applications, John Wiley and Sons, 1962.

7. I. N. Herstein, Topics in Algebra, Blaisdell, 1964.

A-4. Machines, Languages and Algorithms

1. Edward F. Moore, Ed. Sequential Machines: Selected Papers, Addison-Wesley, 1964. Esp. the two papers on state graphs and regular expressions: R. McNaughton and H. Yamada, Regular expressions and state graphs for automata; I. M. Copi, C. C. Elgot and J. B. Wright, Realization of events by logical nets.

2. Arthur Gill, Introduction to the Theory of Finite-State Machines, McGraw-Hill, 1962.

3. Yehoshua Bar-Hillel, M. Perles and E. Shamir. On formal properties of simple phrase structure grammars. Chapters 9 of Bar-Hillel Language and Information, Addison-Wesley and The Jerusalem Academic Press, 1964.

4. Seymour Ginzburg, The Mathematical Theory of Context-Free Languages. McGraw-Hill, 1966. Principally Chapters 1, 2, 3 and 4.

5. T. V. Griffiths and S. R. Petrick, On the relative efficiencies of context-free grammar recognizers. Comm. of the ACM, Vol. 8, No. 5, May 1965, pp 289-300.

6. B. A. Trakhtenbrot, Algorithms and Automatic Computing Machines, D. C. Heath, Boston, 1963.

7. Marvin Minsky, Computation: Finite and Infinite Machines, Prentice Hall, 1967.

References 1 and 2 are the basis for a good development of state diagrams for finite automata and their relation to "regular expressions" (Kleene's theorem). Although highly mathematical and not pedagogical, 3 and 4 are among the best discussions of formal linguistics currently available; 5 is a neatly integrated treatment of parsing procedures. Reference 6 is a brief, informal but well motivated, introduction to computability concepts; Reference 7 is the best available text giving a comprehensive coverage of this subject area.

APPENDIX B

Some Electrical Engineering Curricula with a Concentration in Computer Science

CURRICULUM LEADING TO DEGREE

B. S. - Computer Science Program

College of Engineering
University of California, Berkeley

	quarter hours
FRESHMAN YEAR	
Mathematics	12
Chemistry	12
Physics	7
Electives[1]	14
SOPHOMORE YEAR	
Mathematics	12
Physics	12
Electives[1]	21
JUNIOR YEAR	
Electric Circuits	8
Electronic Circuits	5
Electronics and Circuits Laboratory	6
Linear Systems Analysis	4
Computers and Information Processing	4
Restricted Electives[2], Technical Elective[3], Humanistic-social	18
SENIOR YEAR	
Switching and Computer Circuits	6
Digital Computer Systems	7
Laboratory	2
Restricted Electives[2], Technical Electives[3], Humanistic-social	30
TOTAL	180 qtr hrs

[1]The electives for the freshman and sophomore years include at least 15 hours of humanities or social sciences, plus

Computers and their Applications	4
Introduction to Electronic Systems, Circuits and Devices	4
Engineering Mechanics	4
Properties of Materials	

[2]Restricted Electives: 3 courses from an available list, including mathematics, physics, engineering courses.

[3]Technical Elective: 25 units of upper division computer science, engineering, mathematics, physics, statistics, or other natural science courses.

Humanistic-social: Total in the program must meet minimum College requirements.

TYPICAL CURRICULUM LEADING TO THE DEGREE

S.B. in Electrical Engineering[1]
(Computer Science Program)

Department of Electrical Engineering
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

	credit hours[5]
FRESHMAN YEAR	
Calculus	24
Chemistry	12
Physics	24
Introduction to Automatic Computation	6
Humanities	18
Elective	6
SOPHOMORE YEAR	
Physics[2]	12
Physics	12
Elements, Systems and Computation[2]	12
Elements, Systems and Computation	12
Programming Linguistics[3]	12
Elective[2]	12
Humanities	18
JUNIOR YEAR	
Circuits, Signals and Systems	12
Electromagnetic Fields and Energy	12
Computation Structures[3]	12
Computer Systems[3]	12
Electives[4]	24
Humanities	18
SENIOR YEAR	
Electives[4]	60
Humanities	18
Thesis	12
TOTAL	360 credit hrs[5]

[1]An unofficial curriculum presently under consideration.

[2]Subjects to meet an Institute "science distribution requirement."

[3]Basic computer science courses now under development.

[4]A variety of suitable computer science elective subjects is currently available.

[5]Three credit hours is approximately one semester contact hour.

CURRICULUM LEADING TO DEGREE

B.S. in Computer Science

College of Engineering
UNIVERSITY OF UTAH

	credit hours
FRESHMAN YEAR	
Mathematics	15
Chemistry	10
Freshman University Requirements	17
Introduction to Computer Science	3
SOPHOMORE YEAR	
Physics	15
Mathematics (Engr. Math + Algebra)	8
Circuits (EE)	4
Computer Organization and Programming	4
Humanities and Social Sciences	14
JUNIOR YEAR	
Mechanics	4
Electronics	4
Programming Languages	4
Computer Organization	4
Logical Design	4
Numerical Analysis	4
Thermodynamics	4
Electives (14 Humanities and Social Sciences, 6 Technical)	20
SENIOR YEAR	
Programming Systems	4
Systems (EE)	4
Senior Project	3
Humanities and Social Science Electives	8
Technical Electives	30
TOTAL	186 qtr hrs